Decision Manifolds: Classification Inspired by Self-Organization

Georg Pölzlbauer, Thomas Lidy, Andreas Rauber
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstr. 9–11, Vienna, Austria
email: {poelzlbauer,lidy,rauber}@ifs.tuwien.ac.at

Keywords: Decision Manifolds, supervised learning, ensemble classification

***Abstract*— We present a classifier algorithm that approximates the decision surface of labeled data by a patchwork of separating hyperplanes. The hyperplanes are arranged in a way inspired by how Self-Organizing Maps are trained. We take advantage of the fact that the boundaries can often be approximated by linear ones connected by a low-dimensional nonlinear manifold. The resulting classifier allows for a voting scheme that averages over neighboring hyperplanes. Our algorithm is computationally efficient both in terms of training and classification. Further, we present a model selection framework for estimation of the paratmeters of the classification boundary, and show results for artificial and real-world data sets.

1 Introduction

In this paper, we present a neural classifier algorithm for two-class problems that aims at approximating decision boundaries locally by multiple linear separating hyperplanes. This approximation is performed by placing a representative that describes the decision boundary in its vicinity where the data sample is sufficiently dense. Our classifier, which we call Decision Manifold, subjects the shape of the decision boundary to topological constraints, similar to the output space topology of Self-Organizing Maps. The local classifiers are moved to their appropriate positions along the decision boundaries. We further present a model selection scheme in order to estimate the correct topology of the decision surface. As we show in the experiments section, our classifier is comparable in performance to modern supervised learning algorithms. Apart from fast training, the advantage of our method lies in the exploitation of the classifier topology, which is used during training to align the classifiers to achieve an ordered representation of the decision boundary, and to avoid overfitting and local minima in the placement of the local classifiers. In the last stage of the training phase, when the positions of the individual local classifiers have converged, the topology is used to fine-tune classification performance by determination of the optimal voting weights. The remainder of the paper is organized as follows: In Section 2, we discuss related techniques. In Section 3, we describe essential concepts that our technique is based upon. Section 4 outlines the training algorithm, optimization of classification accuracy by adapting the voting weights of the classifiers in the committee, and the model selection framework. Applications to benchmark supervised learning data sets are given in Section 5, as well as comparisons to state-of-the-art classifiers. Section 6 summarizes our work.

2 Related Work

There has been a lot of effort in the past decades to design piecewise linear classifiers, starting with the work of Sklansky [8]. In this particular approach, a number of hyperplanes is created and the feature space is split into partitions. However, the algorithm has severe complexity problems, as the number of partitions grows exponentially with the number of hyperplanes. Other similar approches include Learning Vector Quantization [4].

A lazy learning algorithm that is comparable to our approach is proposed in [2], where for each pattern k training samples from its vicinity are selected. A classifier is trained on this subset of the data. The results are good in terms of accuracy, but this approach is computationally expensive, as a new classifier has to be trained with every pattern to be classified. Our algorithm is similar in terms of accuracy, but is not a lazy learner as it takes advantage of local representatives that cover an area of the feature space, thus not suffering from these performance issues.

As our classifier consists of several local classifiers that perform the actual classification task by a voting scheme, ensembles of classifiers are related to our technique. Mixtures of local experts [6] train a committee of multi-layer perceptrons in a competitive learning approach where each MLP specializes on a region of the feature space. The classification is performed by a selector that assigns the samples to the local experts.

Our method relies heavily on the concepts of topology and local linear approximation. Manifold learning methods, such as ISOMAP [9] and Local Linear Embedding [7], follow a similar approach, but are mainly used for nonlinear dimensionality reduction and projection. Our goal is to approximate the decision boundaries rather than the data points, but we use similar techniques like local approximation as manifold learning.

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

3 Elementary Concepts

In this section, we briefly outline several concepts that our algorithm is based upon and which will serve as building blocks. The first one is the Batch SOM algorithm for unsupervised learning, for the details of which beyond the material presented here please refer to [3]. The SOM is computed by presenting an unlabeled data set consisting of vectors $\mathbf{x}_i \in \Re^D$ to M prototype vectors $\mathbf{m}_j \in \Re^D$, that are usually aligned along a two-dimensional grid topology. The proptotype vectors $\mathbf{m}_j$ are updated for T epochs, such that they both represent the original data in a vector quantization sense and preserve the grid topology. In this paper, we use output spaces of arbitrary dimensions rather than only two dimensional ones, which are most suitable for visualization purposes [5]. We consider $\mathfrak{M}$-dimensional topologies, where the prototype vectors are aligned in an equidistant and rectangular way. We refer to such a topology as $\{d_1 \times d_2 \times \cdots \times d_{\mathfrak{M}}\}$, where $d_1, \ldots, d_{\mathfrak{M}}$ are the number of nodes per output space axis. The pairwise distances in output space between the unit associated with $\mathbf{m}_i$ and $\mathbf{m}_j$ are stored in the symmetric topology matrix A. In order to preserve this topology, the neighborhood kernel K is introduced that determines the mutual influence of two prototype vectors onto each other based on their output space distance. The most common choice for the neighborhood kernel is the Gaussian function, defined as $K_{\sigma(t)}(i,j) = \exp\{-\frac{A_{ij}}{\sigma(t)^2}\}$, where $\sigma(t)$ is a monotonously decreasing function that defines the width of the kernel. Other than the sequential version, the Batch SOM algorithm does not depend on the ordering of the data samples, and does not require a learning rate parameter $\alpha(t)$. It is computed by first assigning the samples to their closest prototype vector, and computing centroids for each partition:

$$I(\mathbf{x}) = \arg \min_{j \in \{1,\ldots,M\}} \|\mathbf{x} - \mathbf{m}_j\| \tag{1}$$

$$\mathfrak{S}_j = \{i | \mathbf{x} \in X \wedge I(\mathbf{x}_i) = j\} \tag{2}$$

$$\mathbf{n}_j = \frac{1}{|\mathfrak{S}_j|} \sum_{\mathbf{x}_i \in X_{\mathfrak{S}_j}} \mathbf{x}_i \tag{3}$$

where $\|\cdot\|$ denotes Euclidean distance, and $I(\cdot)$ represents the index of the prototype vector the data point $\mathbf{x}$ is assigned to. Function $I(\cdot)$ together with the prototype vectors defines a Voronoi tessellation of the feature space. The set of indices of samples mapped to prototype $\mathbf{m}_j$ is denoted as $\mathfrak{S}_j$. For referring to the subset of samples belonging to prototype vector j, we write $X_{\mathfrak{S}_j}$. $|\cdot|$ denotes the cardinality of a set, i.e. in this case the number of data points assigned to a prototype vector. $\mathbf{n}_i$ is the center of the data points assigned to node i. The update step of epoch t is performed by calculating the new prototype vectors $\mathbf{m}_j$ with the Batch SOM training algorithm:

$$\mathbf{m}_j := \frac{\sum_{k=1}^{M} |\mathfrak{S}_k| \cdot K_{\sigma(t)}(k,j) \cdot \mathbf{n}_k}{\sum_{k=1}^{M} |\mathfrak{S}_k| \cdot K_{\sigma(t)}(k,j)} \tag{4}$$

The next essential concept is linear classification. The Decision Manifold method is designed to work with any kind of classifier that results in a single separating hyperplane that can be represented as a vector in homogenous coordinates. For the sake of brevity and simplicity, we limit our discussion to the Moore-Penrose Pseudoinverse (MPI). Other than in the unsupervised setting, data samples $\mathbf{x}_i \in \Re$ are labeled with binary class $y \in \{-1, +1\}$. The whole data set is written as matrix X, where rows correspond to patterns and columns to features, and a vector of labels $\mathbf{y}$. The MPI computes the separating hyperplane in a single step:

$$\tilde{\mathbf{w}}_{\text{MPI}}(X, \mathbf{y}) = (\tilde{X}^\top \tilde{X})^{-1} \tilde{X}^\top \mathbf{y} \tag{5}$$

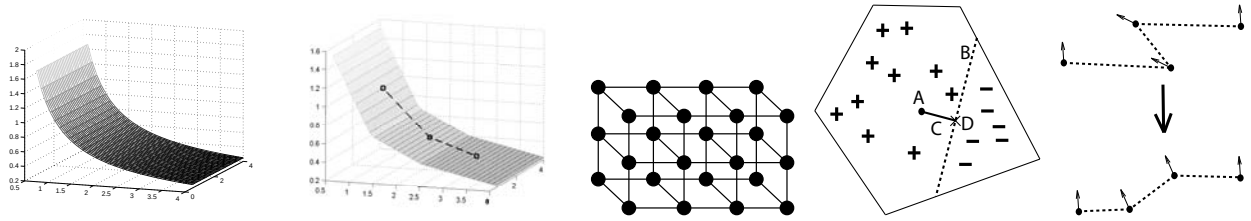
After training the classifier, we obtain the normal vector[1] $\tilde{\mathbf{w}}(X, \mathbf{y})$ to the separating hyperplane pointing in the direction of class $+1$. Using this notation, classification of datum x is performed by estimating label $\hat{y}$.

Finally, some considerations on the topology and shape of decision surfaces of general classifiers are given. Every classification algorithm explicitly or implicitly performs an estimation of such a decision surface which partitions the feature space into disjoint regions that are assigned a label. Mathematically, a decision surface is a hypersurface (i.e. of dimension $D-1$, and of arbitrary shape). We assume this decision boundary to consist of a finite number of topological manifolds, thus the possibly non-contiguous hypersurface can be decomposed into contiguous subsets. Bounded topological manifolds, i.e. ones that do not extend to infinity, can be categorized according to their dimensionality. For example, a line segment is topologically equivalent to an arc of a circle since both are one-dimensional manifolds. Further, a topological manifold has a surface that is locally Euclidean and can thus be approximated by a patchwork of hyperplanes.

Fig. 1(a) shows a possible decision boundary in three dimensions of the form $f(x,y) = 1/x$. Fig. 1(b) shows a linear approximation of this surface by three hyperplanes aligned along a one-dimensional manifold, i.e. a curve. The approximation shows one important concept that we exploit: As the function value is constant along the y coordinate, the patchwork can be aligned along a one-dimensional manifold. There is no need to introduce additional hyperplanes along the y-coordinate as this is already covered by the linearity of the hyperplanes that extend to infinity in the y direction. We can thus represent the two-dimensional decision manifold by a one-dimensional topology, along which the hyperplanes are patched together. If their number is increased, the manifold can be approximated at arbitrary precision. In Section 5, our experiments show that the required topology dimension is usually very much lower than the feature space dimension. Fig. 1(b) also gives an outline of the goal of our method:

[1] Homogeneous coordinates, which include a bias-term in the first coordinate of the vector are denoted by a tilde.

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de



(a) (b) (c) (d) (e)

Figure 1: (a) Decision hypersurface in 3-dimensional space, (b) approximation by 3 local hyperplanes aligned along a 1-dimensional manifold, (c) Graph of $\{4 \times 3 \times 2\}$ Topology, (d) training of a linear classifier; thin lines indicate borders of Voronoi set, "+" and "-" denote samples, "A" center point $\mathbf{n}$, "B" separating hyperplane, "C" normal $\mathbf{v}'$, "D" projected center: $\mathbf{c}'$, (e) smoothing over neighborhood topology: upper part is before, the lower part after the smoothing step

A set of points that represent the center of a classifier hyperplane (square markers) and their topologically correct order (dashed lines). Following this motivation, the algorithm we describe in the next section aims to estimate a local approximation of the decision hypersurface with the following characteristics:

- The decision surface is estimated reliably where data density is sufficiently high, otherwise extrapolation is used by extending the decision hyperplanes to infinity.
- The decision surface can be approximated locally by linear classifiers that are ordered along a topology that is as low-dimensional as possible.

Our classifier consists of a set of local linear classifiers that are subjected to a given topology. We refer to the classifier's topology as the number of local linear classifiers along the axis in each dimension; the topology of the classifier in Fig. 1(b) would be $\{3\}$. An example of a topology with $\{4\times3\times2\}$ local classifiers is shown in Fig. 1(c) where each dot at the intersections of the grid's edges represents a local linear classifier. These are similar to the units of a Self-Organizing Map of according dimensionality, i.e. in this case a 3-dimensional SOM. However, the major difference to SOMs is that prototype vectors are placed where data density is high, and our method places hyperplanes between dense areas of samples with different labels.

Algorithm 1 Training Algorithm

```
randomly initialize c_j and v_j
for t = 1 to T do
    for j = 1 to M do
        find set of samples S_j assigned to classifier j
        compute weight γ_j
        if w_j > 0 then
            compute separating hyperplane w̃_j
            compute projection c'_j of center point n_j to w̃_j
            compute normal v'_j from w̃_j
        end if
    end for
    compute new c_j according to A, c', and σ(t)
    compute new v_j according to A, v', and σ(t)
end for
compute σ_final
```

4 Decision Manifolds

For supervised learning the training samples $\mathbf{x}_i$ are each associated with a class label $y_i \in \{-1, 1\}$. The Decision Manifold consists of M local classifiers, each specified by a pair of a representative $\mathbf{c}_j \in \mathbb{R}^D$ and a classification vector $\mathbf{v}_j \in \mathbb{R}^D$ that is orthogonal to the decision hyperplane of the local classifier. Both $\mathbf{c}_j$ and $\mathbf{v}_j$ are initialized randomly. $\mathbf{c}_j$ determines the position of the classifier in feature space, while $\mathbf{v}_j$ is relevant for performing the classification. Samples to be classified are assigned to their closest representative and classified by the local hyperplane defined by $\mathbf{v}_j$. The goal of the training algorithm is to put the representatives $\mathbf{c}_j$ in the adequate positions and to let $\mathbf{v}_j$ point in the correct direction for classification. When training has finished, the Decision Manifold is described by the tuple $\{C, V, \sigma_{\text{final}}, A\}$, where A is the given topology, C and V are the sets of representatives and classification vectors, respectively, and σ_{final} is the optimal classification width, which will be defined later in this section. An example of a trained classifier can be seen in Fig. 2(g). The thin lines delimit the areas where samples are assigned to the respective classifier, and thick lines and arrows represent the separating hyperplane. The dashed lines refer to the topology of the Decision Manifold. Note that the connected classifiers are actually next to each other. This ordering is induced by the imposed one-dimensional topology in this example, which will be explained in the course of this section. We further provide an iterative algorithm that aims at placing the representatives in a way such that the predefined topology of the classifiers is preserved, i.e. that neighboring classifiers are responsible for neighboring areas of the data space. The adjacency matrix A that defines the topology, and thus the shape of the decision boundary, is assumed to be given as a parameter. Select-

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

ing a suitable topology will be explained in greater detail later in this section.

Training is performed for a predefined number of epochs T. Our experiments have shown that this parameter is non-critical to the performance of our classifier, and we assume $T = 5$ in the rest of this paper. An outline of the training algorithm is given in Algorithm 1, and we will refer to the line numbers when explaining each step. In the first step of each epoch, the data samples are assigned to the closest local classifier representative (line 4), as in Equs. (1) and (2). Again, the set of indices of samples mapped to the j^{th} local classifier is denoted as $\mathfrak{S}_j$, and the sets of samples and labels assigned to it as $X_{\mathfrak{S}_j}$ and $\mathbf{y}_{\mathfrak{S}_j}$, respectively. Note that other than in the case of the unsupervised SOM, the representatives $\mathbf{c}_j$ are not true prototype vectors placed where data density is high, but rather in positions where there is a transition between two neighboring areas of different classes, which will be explained in detail in the next paragraphs. Further, we are interested in the centers $\mathbf{n}_j$ of the partitions, which do not necessarily coincide with $\mathbf{c}_j$.

Once all the samples have been assigned, a linear classifier is trained for each partition to obtain a separating hyperplane. Since this can only be performed if samples of both classes are present, and the number of training samples will be of interest in a later step, we compute the weighting factor (line 5) as follows:

$$\gamma_j = \begin{cases} |\mathfrak{S}_j| & \text{if } (-1 \in \mathbf{y}_{\mathfrak{S}_j}) \wedge (+1 \in \mathbf{y}_{\mathfrak{S}_j}) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

This means that classifier positions that are located in areas of high data density and contain data samples of both classes will receive higher weights. If $\gamma_j > 0$, we train a linear classifier (line 7) on the subset $X_{\mathfrak{S}_j}$ from which we can extract the separating hyperplane $\tilde{\mathbf{w}}_j(X_{\mathfrak{S}_j}, \mathbf{y}_{\mathfrak{S}_j})$. This hyperplane must pass through the convex hull of the training samples $X_{\mathfrak{S}_j}$ and thus partly lies within the Voronoi region of representative $\mathbf{c}_j$. We are interested in updating the representative $\mathbf{c}_j$ such that it lies on the separating hyperplane and does not drastically change the Voronoi partition for consecutive epochs. As all the points that lie on the hyperplane are equivalent to describe it along with the normal vector, we compute a new preliminary representative $\mathbf{c}'_j$ that lies on the hyperplane by projection of the data centroid $\mathbf{n}_j$ and store the information for classification in the normalized vector $\mathbf{v}'_j$:

$$\mathbf{c}'_j = \pi_{\tilde{\mathbf{w}}_j}(\mathbf{n}_j), \quad \mathbf{v}'_j = \frac{\mathbf{w}_j}{\|\mathbf{w}_j\|} \tag{7}$$

where $\pi_{\tilde{\mathbf{w}}}(\cdot)$ denotes orthogonal projection onto the hyperplane specified by $\tilde{\mathbf{w}}$. This projection ensures that the representative is placed on the decision boundary and is the point closest to the centroid of the Voronoi partition. A schematic overview of this update step (from partitioning in Voronoi sets to calculation of $\mathbf{c}'_j$ and $\mathbf{v}'_j$) is shown in Fig. 1(d). For the set of points delimited by the thin lines that represent the borders of the Voronoi region, the separating hyperplane obtained by linear classification is shown as a dashed line ("B"). The centroid $\mathbf{n}_j$ ("A") is projected along the normal ("C") to its position $\mathbf{c}'_j$ ("D").

At this point, the topology that puts the representatives into relation comes into play. As the local classifiers should be ordered such that they resemble a continuous decision surface, a smoothing step is performed that takes care of ordering the local representatives to obey the topology induced by matrix A. This step is very similar to the update process used in the training of SOMs. We use the neighborhood kernel weighted by the number of samples in each Voronoi partition to calculate smoothed versions along the topology of both the classification vectors $\mathbf{v}_j$ and representatives $\mathbf{c}_j$. The idea behind this is making them more similar to their topological neighbors in order to achieve a smooth representation of the decision boundary and to avoid overfitting. Using a formula similar to the Batch SOM training defined in Equ. (4), the updated local classifiers (lines 12, 13) are

$$\mathbf{c}_j = \frac{\sum_{k=1}^{M} K_{\sigma(t)}(k,j) \cdot \gamma_k \cdot \mathbf{c}'_k}{\sum_{k=1}^{M} K_{\sigma(t)}(k,j) \cdot \gamma_k}, \tag{8}$$

$$\mathbf{v}_j = \frac{\sum_{k=1}^{M} K_{\sigma(t)}(k,j) \cdot \gamma_k \cdot \mathbf{v}'_k}{\sum_{k=1}^{M} K_{\sigma(t)}(k,j) \cdot \gamma_k} \tag{9}$$

The new representatives $\mathbf{c}_j$ and classification vectors $\mathbf{v}_j$ are derived from the preliminary versions $\mathbf{c}'_j$ and $\mathbf{v}'_j$ defined in Equ. (7) and subjected to the kernel smoothing. Further, $\mathbf{c}_j$ and $\mathbf{v}_j$ are weighted according to topological distance and by the number of data points γ_j they represent. Thus, a local classifier that represents many data points will pull its neighbor that represents relatively few samples in its direction. Also, an effect known in vector quantization as magnification factors [10] occurs that concentrates representatives in dense areas. During the first few epochs, a high value of σ ensures that the local classifiers are aligned according to the topology. As σ declines, the local classifiers specialize and reach their final positions. The influence through topological proximity thus vanishes. Fig. 1(e) explains how this smoothing happens over a possibly overtrained situation, and shows how the representatives and classification vectors are realigned to be more similar to their neighbors. After this step, the current epoch is finished.

Fig. 2(a)–(h) shows an example of training with 5 local classifiers with a one-dimensional topology, referred to as $\{5\}$, on a simple non-linearly separable data set. It consists of 200 samples that are distributed along a sine-wave with Gaussian noise, where the class "+" is offset by a small vertical margin. In the figures, data points are represented as "+" and "o". The classification vectors $\mathbf{v}_j$ are shown as the arrows pointing to the direction of the "+" class, and the hyperplanes as thick, solid lines. The positions of the representatives $\mathbf{c}_j$ are denoted as small squares where hyperplane and classification vector intersect.The topology of

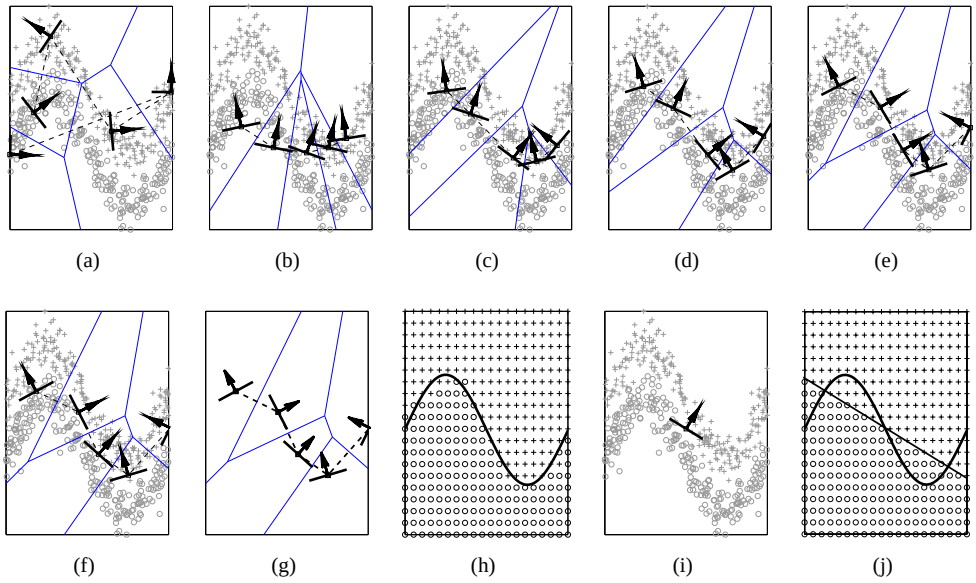


Figure 2: Training algorithm on non-linearly separable data set: (a) after initialization, (b)–(f) after 1st–5th epoch, (g) after training (without data samples), (h) classification results and Bayes optimal decision boundary, (i) only one classifier, (j) classification with one classifier

the local classifiers is visualized by dashed lines connecting adjacent local classifiers (i.e. where $A_{ij} = 1$). In Fig. 2(a), the randomly initialized Decision Manifold is shown. After the first training epoch, depicted in Fig. 2(b), the local classifiers are arranged in a more orderly fashion as a result of smoothing according to Equ. (8), yet do not classify well after this iteration. Figs. 2(c)–(e) show the consecutive stages of training as σ, the parameter controlling the mutual influence between topological neighbors, is decreased. The purpose of the earlier epochs is to roughly align the representatives along the topology, while the later epochs are for fine-tuning the individual areas of each classifier. The finished Decision Manifold is shown in Figs. 2(f),(g) with and without the data points. It can be seen in Fig. 2(h), which shows the Bayes decision boundary as a thick line along with the decisions over the whole feature space, that the optimal decision boundary has been approximated very reliably. Figs. 2(i),(j) show a trained Decision Manifold of topology $\{1\}$ that consists of only a single classifier, and is thus equivalent to performing a simple linear classification. This topology is not capable of approximating the decision boundary sufficiently. Selecting the (unknown) correct topology in the first place is thus very important and will be dealt with later in this section.

The complexity of the training algorithm can be calculated as the sum of the sample assignment to representatives $O(N \cdot M)$, training of the M linear classifiers $O(M \cdot O_l(N))$, where O_l denotes the classifier complexity, and the complexity of the SOM update step $O(M^2)$. For T epochs, this results in $O(T \cdot (N \cdot M + M \cdot O_l(N) + M^2))$. The training algorithm can be implemented very efficiently. In our current experiments running on a 1.60 GHz computer and a Matlab implementation of the algorithm, the training duration for a data set with $N = 400$ and $D = 50$, and training with $T = 5, M = 20$ is less than a second.

When the local classifiers are in their final positions, the training algorithm enters its last phase where the classification performance is fine-tuned on the training data set. In its most simple form, classification is performed by assigning a data sample to its closest linear classifier in the same way as during training, and then classifying it according to its position relative to the hyperplane, and is defined as:

$$\hat{y}(\mathbf{x}) = \text{sign}\big((\mathbf{x} - \mathbf{c}_{I(\mathbf{x})})^\top \cdot \mathbf{v}_{I(\mathbf{x})}\big) \qquad (10)$$

A more sophisticated approach takes advantage of the topology: Since neighboring representatives are expected to form a smooth decision boundary, voting of the local classifier ensemble according to the topological proximity of every classifier to the representative closest to $\mathbf{x}$, i.e. $\mathbf{c}_{I(\mathbf{x})}$, can significantly increase accuracy:

$$\hat{y}(\mathbf{x}, \sigma) = \text{sign}\Big(\sum_{j=1}^{M} K_\sigma(I(x), j) \cdot \hat{y}(\mathbf{x})\Big) \qquad (11)$$

where σ is the smoothing width, with higher values increasing the influence of distant classifiers in terms of the topology. Effectively, the datum $\mathbf{x}$ is classified by all the hyperplanes, and the final decision is performed as weighted voting where neighboring classifiers receive a higher weight. In the last step of the algorithm (line 15), the training set accuracy (the percentage of correctly classified samples) of

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

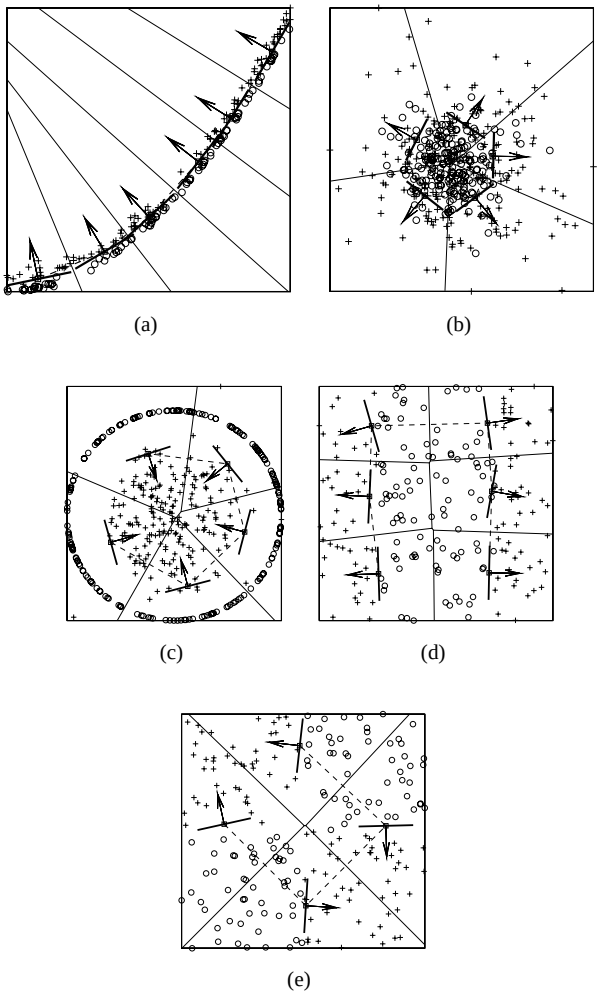


Figure 3: Artificial Datasets: (a) quadratic decision surface, (b) 2 Gaussians with different variances, (c) ring and Gaussian, (d) 2 linear separations, (e) XOR

the classifier is maximized with respect to σ. In our experiments, optimizing σ was responsible for a gain in accuracy of up to 3%. This is done by sampling several values of $0 < \sigma \ll M$ and we set

$$\sigma_{\text{final}}(C, V, X, \mathbf{y}) = \arg\max_{\sigma} acc(C, V, X, \mathbf{y}, \sigma) \quad (12)$$

Next, we discuss five data sets representing typical nonlinearly separable supervised learning problems, each consisting of 200 samples. They are shown in Fig. 3 together with the trained Decision Manifold. Note that the dashed lines again do not determine the classification boundaries, but indicate the topological neighborhood of the local classifiers.

The first example, shown in Fig. 3(a), consists of samples divided by a boundary along a quadratic polynomial. The Bayes decision boundary is nonlinear in this case. 6 local linear separators are trained, resulting in the ordered approximation visualized in the figure.

In Fig. 3(b), both classes are normally distributed with the same center, but class "+" has a higher variance. In this case, there is considerable overlap between in all regions. The Bayes optimal boundary runs along a circle where the class conditional probability density functions intersect. As Fig. 3(b) shows, our algorithm is capable of finding this border. Within this circle, there will likely be some misclassifications of class "+", but class "o" is still more common there.

In a related example shown in Fig. 3(c), class "o" is distributed uniformly on a circle, and class "+" is normally distributed around its center. A Bayes optimal classifier would assign label "o" to any point on the circle and "+" otherwise. The Decision Manifold consists of 5 local classifiers along a one-dimensional bounded topology. This is a mismatch to the topology of the actual decision boundary, which is also one-dimensional but circular. However, most of the classification boundary can still be captured. Due to the sparsity of samples outside the circle, the decision boundary does not identify these samples correctly as "+".

Fig. 3(d) shows the results for a data set where the Bayes decision boundary is not contiguous. The decision hypersurface is split into two linear manifolds (parallel lines). After training of 6 local classifiers with a one-dimensional topology, the result shows that our method is capable of dealing with this problem. It could have been solved with only 2 linear classifiers, but we chose it as a demonstration of how the representatives are aligned in case of a topology breach. Further, we want to demonstrate that the Decision Manifolds' performance does not deteriorate in case of over-specification in terms of the number of classifiers.

In Fig. 3(e), separation of the XOR-problem is demonstrated with four local classifiers. Here, the feature space is split along the diagonals, and the neighborhood relations are disregarded for classification, i.e. σ_{final} is close to zero and no voting is performed.

It has been mentioned above that selection of a suitable topology is critical for the performance of our algorithm, as the topology constrains the shape of the decision boundary. Here, we present a scheme for topology estimation, and a model selection approach that trains several Decision Manifolds with different topologies and selects the best one. Training over a number of topologies can be afforded due to the computational inexpensiveness of training a Decision Manifold. The topology connecting the representatives can have at most dimension $D - 1$. If we assume that each axis of the topology grid contains 5 local classifiers, and the dimension of the data set is 50, this would result in 5^{49} classifiers, which clearly can not be handled. As the dimension of the topology is likely to be much lower, we have to perform a reasonable estimate of the intrinsic dimensionality of the data set. This task has been addressed previously [1], but we apply a simple PCA-guided scheme. From the data set X, we extract the ordered set of eigenvalues $\lambda_1, \ldots, \lambda_D$, which is normalized to add up to one. Note that we do not use PCA for dimensionality reduction of the

Table 1: Eigenvalues of PCA, Dimensionality estimation for the topology connecting the local classifiers

i	λ_i	Topology for $d_{\text{max}} = i$	Topology without 0,1	Topology Graph
1	0.2618	$\{10\}$	$\{10\}$	
2	0.2164	$\{5 \times 4\}$	$\{5 \times 4\}$	
3	0.1287	$\{4 \times 3 \times 2\}$	$\{4 \times 3 \times 2\}$	
4	0.1094	$\{3 \times 3 \times 1 \times 1\}$	$\{3 \times 3\}$	
5	0.0953	$\{3 \times 2 \times 1 \times 1 \times 1\}$	$\{3 \times 2\}$	
6	0.0853	$\{2 \times 2 \times 1 \times 1 \times 1 \times 0\}$	not valid	
7	0.0525	$\{2 \times 2 \times 1 \times 1 \times 1 \times 0 \times 0\}$	not valid	
8	0.0506	$\{2 \times 2 \times 1 \times 1 \times 0 \times 0 \times 0 \times 0\}$	not valid	

data set, just for estimation of the topology. For the approximate desired number m of classifiers distributed along all discrete axes, we construct the topology as follows:

$$\mathfrak{d}_i = \left\lfloor \frac{m \cdot \lambda_i}{\sum_{j=1}^{d_{\text{max}}} \lambda_j} \right\rfloor \quad , \tag{13}$$

where $\mathfrak{d}_i$ is the number of classifiers along the rectangular topology's i^{th} axis, and $\lfloor \cdot \rfloor$ denotes rounding down to the closest integer. The topology is then $\{\mathfrak{d}_1 \times \cdots \times \mathfrak{d}_{d_{\text{max}}}\}$. d_{max} is the dimension of the topology, at most the data set dimension D. $\mathfrak{d}_i = 1$ can be omitted, since an axis that only holds one discrete coordinate does not provide any information, and if any $\mathfrak{d}_i = 0$, the topology is not valid. We iteratively construct the topology as a ratio of the up to D eigenvalues $\lambda_1 : \lambda_2 : \cdots : \lambda_{d_{\text{max}}}$. For example, the Pima Indian Diabetes data set, which will be discussed in the next section, consists of 8 variables. Its ordered eigenvalues are $0.26, 0.21, 0.12, 0.10, 0.09, 0.08, 0.05, 0.05$; for $m = 10$ the candidate topologies are shown in Table 1. We repeat this procedure by iterating $m = 1 \ldots 10$ and obtain a set of 17 distinct topologies (the smallest of which is $\{1\}$, the largest is $\{4 \times 3 \times 2\}$). We chose 10 as a reasonable upper limit for m as real-world problems rarely require large numbers of separating hyperplanes. For estimation of the performance of the Decision Manifold with each topology, we split X into training and test set. After each classifier has been trained, σ_{final} is estimated on the training set. The resulting Decision Manifolds are then evaluated on the test set, and the best model is selected.

For estimation of the generalization accuracy, we perform 10-fold cross-validation. The remaining 90 % of the data set are divided into training and test set by the ratio 80 to 20. The topologies for the models are then estimated by the PCA approach described in the previous paragraphs and the models are trained, and σ_{final} is estimated. The resulting classifiers are then evaluated on the test set, and the best model is selected for validation, resulting in the final accuracy measurement. This Training-Test-Validation approach is summarized in Table 2 for one fold.

To summarize the methodological part of this paper, we recapitulate the properties of our method:

Table 2: Model Selection

Training			Test	Validation
Classifier	Topology	Estimate σ	Test Set Accuracy	Validation Set Accuracy
1		0.27	87 %	
2		1.19	88 %	Best Model 85 %
3		2.14	77 %	
4		0.66	82 %	
. . .				

- The algorithm is a stochastic supervised learning method for two-class problems.
- Computation is very efficient.
- The topology induced by adjacency matrix A defines the ordering and alignment of the local classifiers; it is also exploited for optimizing classification accuracy by a weighted voting scheme.
- As the topology of the decision hypersurface is unknown, we apply a heuristic model selection that trains several classifiers with different topologies.
- The classifier performs well in case of multiple non-contiguous decision surfaces and non-linear classification problems such as XOR.

5 Experimental Results

The experiments are performed on 7 supervised learning benchmark data sets taken from the UCI Machine Learning Repository[2]: Bupa Liver Disorders, Pima Indian Diabetes, Spam, Ionosphere, Statlog Heart Disease, Sonar, and Statlog German Credit data bases. In case of categorical features, 1-to-N encoding has been applied, otherwise the data has been normalized with a zero-mean-unit-variance

[2] Available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

Table 3: Comparison of average 10-fold cross-validation error (in %)

Data	Samples	Dim.	Topol.	Dec. Mf.	lin. SVM	pol. SVM	RBF SVM	R. F.	Dec. Tr.	k-NN
Bupa	345	6	$\{5 \times 2\}$	29.4	29.6	40.0	30.1	**27.2**	29.3	33.9
Pima	768	8	$\{3 \times 2\}$	**20.9**	23.3	25.3	23.0	23.3	25.5	25.0
Spam	4601	57	$\{7\}$	7.7	7.2	21.7	6.8	**4.7**	8.6	9.2
Iono	351	34	$\{3\}$	12.8	11.7	12.8	**6.0**	6.3	14.0	14.8
Heart	270	13	$\{4\}$	18.5	**17.0**	17.8	18.5	20.0	19.6	18.1
Sonar	208	60	$\{3 \times 3\}$	**14.3**	26.4	18.3	16.3	14.8	29.3	16.8
Credit	1000	20	$\{5\}$	26.5	24.8	26.6	**23.1**	**23.1**	27.1	26.5

transformation. The number of samples and dimensions are summarized in the first columns of Table 3.

We compare the results of the Decision Manifolds to Random Forests, linear, polynomial, and radial basis function SVMs, k-Nearest Neighbors, and Decision Trees, by averaging the out-of-bag error for 10-fold crossvalidation. The experiments of these classification algorithms have been performed with R, a computational statistics environment [3]. The parameters have been estimated according to the following model selection schemes (separately for each fold) by sampling the parameter in question in 15 steps: For polynomial SVMs, the degree of the kernel has been tuned in the range of 2–5, and for RBF kernels, the radius has been tuned. k-NN was performed with $1 \leq k \leq 29$, where k are only the odd numbers. For Random Forests, 500 trees have been trained and random splitting with 30% of the variables. In this case, no model selection has been performed, as Random Forests are generally very robust with respect to the choice of parameters. For Decision Trees, different pruning thresholds have been tried, a parameter that influences the size of the tree grown. Decision Manifolds have been trained with $T = 5$ epochs, and with Moore-Penrose-Inverse as underlying linear classifier.

Table 3 summarizes the results. In column "Topol.", an example of the most frequently selected topology is shown (as this may vary over different folds, we chose to depict a typical one). Decision Manifolds outperform the other algorithms on the Pima and Sonar data sets. Random Forests and linear SVMs each perform better than our method on 4 of the 7 data sets. k-NN, Decision Trees, and polynomial SVMs perform worse in most cases. RBF SVMs are better on 3, and worse on 3 data sets. Overall, there is no algorithm that outperforms every other on all the data sets, and our technique yields good classification results.

In terms of training and classification time, the SVM models and k-NN are significantly slower at higher sample counts then the remaining algorithms (on the Spam data set, training took almost between one and two hours for polynomial SVM, RBF SVM, and k-NN, versus 5 to 15 minutes for the rest of the models).

[3] R can be obtained at http://cran.r-project.org.

6 Conclusion and Future Work

We have proposed a classifier for binary problems by local approximation of decision boundaries of a given topology. We have shown how it can be used to fit various low-dimensional non-trivially separable data sets. For selection of the topology, we have proposed a model selection scheme. Empirical evaluations have shown that Decision Manifolds perform comparable to modern classifiers. Future work will include investigation of topological constraints, such as non-rectangular topologies, and extension to multi-class problems.

References

[1] H. Bauer and T. Villmann. Growing a hypercubical output space in a self-organizing feature map. *IEEE Trans. on Neural Networks*, 8(2):218–226, 1997.

[2] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4:888–900, 1992.

[3] T. Kohonen. The Self-Organizing Map. *Neurocomputing*, 21:1–6, 1998.

[4] Teuvo Kohonen. Improved versions of learning vector quantization. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'90)*, pages 545–550, 1990.

[5] G. Pölzlbauer, M. Dittenbach, and A. Rauber. Gradient visualization of grouped component planes on the som lattice. In *WSOM'05*, pages 331–338, 2005.

[6] R.Jacobs and M. Jordan. Hierarchical mixtures of experts. *Neural Computation*, 6:181–214, 1994.

[7] L. Saul and S. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *J. of Machine Learning Research*, 4:119–155, 2003.

[8] J. Sklansky and L. Michelotti. Locally trained piecewise linear classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2(2):101–111, 1980.

[9] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[10] T. Villmann and J. Claussen. Investigation of magnification control in Self-Organizing Maps and Neural Gas. *Neural Computation*, 18(2):446–469, 2006.